

ALL-IN-1 Information Update
May 1991

Order Number: AD-ND93B-B3

ALL-IN-1 Information Update

Published by
Corporate User Information Products (MRO1-3/LP7)
Digital Equipment Corporation
P. O. Box 1001
Marlborough, MA 01752-9840

The *ALL-IN-1 Information Update* is a quarterly publication reporting on important ALL-IN-1 Software Performance Reports (SPRs), management, and tuning. It also describes workarounds and provides customers with information that may help them avoid problems. Much of the material is developed from SPR answers significant to the general audience and is printed here to supplement the maintenance updates.

Distribution

The *ALL-IN-1 Information Update* is directed to one software contact for each software product. No mailing will be made to addresses without a software contact name. **Address change requests should be sent to the nearest Digital field office. Include the new address and mailing label from the most recently received publication.**

The prerequisite to using Digital Software is the appropriate license. The standard Terms and Conditions and Digital Business Agreement (DBA) contain the license terms for all software other than DECsystem-10.

Susan A. Thompson, Editor

Copyright © Digital Equipment Corporation 1991. All Rights Reserved.

The material in this document is for informational purposes only. Digital believes the information in this publication is accurate as of its publication date; such information is subject to change without notice. Digital is not responsible for any inadvertent errors. Comments on the contents of this publication should be directed to your local Digital field office.

The following are trademarks of Digital Equipment Corporation: ALL-IN-1, CDA, DATATRIEVE, DDIF, DEClaser, DECnet, DECsystem, DECtrace, DECUS, DECUS logo, DTIF, FMS, ULTRIX, VAX Notes, VAXcluster, VMS, WPS-PLUS, and the DIGITAL logo.

CONTENTS

	Page
INTRODUCTION	1
APPLICATION PROGRAMMING	
Programming Update	3
Programming an Application for Performance	5
Programming for Optimal Field Access	11
SYSTEM MANAGEMENT	
Managing an Application for Performance	15
Problem in X.400 Addresses with Multiple Organizational Units	19
Cumulative Index	23
Attention U.S. Software Contract Customers Submitting Software Problems	31
Digital Equipment Computer Users Society (DECUS)	33
DECUS Subscription Service	37
Reader's Comments	39

INTRODUCTION

Welcome to the May 1991 issue of the *ALL-IN-1 Information Update*. The *ALL-IN-1 Information Update* is a quarterly publication that aims to keep you up-to-date with ALL-IN-1 and contains useful information for system managers, application programmers, and ALL-IN-1 users.

Some of the articles for the *ALL-IN-1 Information Update* are developed from reports of problems that we have received from customers and are included to help you to avoid these problems. Other articles contain practical information about ALL-IN-1 features of which users might not be aware.

If you have any questions about ALL-IN-1 Versions 2.3 or 2.4 and are covered by a service contract, contact your local Digital Customer Support Center.

We are eager to provide you with helpful information. Therefore, please take the time to complete the Reader's Comments page at the back of this issue. This will help us get a better idea about the type of articles you would like to see in future issues.

APPLICATION PROGRAMMING

Programming Update

Listing the Trace Log File

This article describes some minor changes that were made in ALL-IN-1 Version 2.4 and are not documented elsewhere.

It is now possible in ALL-IN-1 Version 2.4 to list the trace log file without exiting from ALL-IN-1. Do this as follows:

1. Enter the following commands:

```
<OA$TRA_SET OFF,LOG  
<LIST A1TRACE.LOG
```

2. When you return to the menu, resume tracing to the log file by entering the following command:

```
<OA$TRA_SET LOG
```

Compiling Forms with Unresolved Special Symbols

The ALL-IN-1 forms compiler was amended so that it now reports an error if it cannot resolve one of the following types of symbol:

- Menu option symbols, for example:

```
;;OA$_MO_NICKNAMES;;
```

- Field qualifier symbols, for example:

```
/HARD = OA$_UD_HRD_INDEX_OPTNS  
/LANGUAGE = OA$_LANG_YES_OR_NO
```

When you compile a form library, use the following commands to produce a list of any unresolved symbols, indicating, for each symbol, on which form the symbol was found:

```
OA$TRA_SET IVP,LOG  
OA$FBT_WRITE_LIBRARY formlibrary  
OA$TRA_SET OFF,LOG
```

In addition, if you turn on Installation Verification Procedure (IVP) tracing and log file tracing during the compilation, error messages are written to A1TRACE.LOG.

NOTE

Currently, there is a problem with this feature in relation to forms linked with a .MORE named data directive. If the unresolved symbol is used on a form specified in a .MORE directive, the error is not reported against that form, but against the form that references it.

Programming an Application for Performance

This article gives guidelines on how to design and write applications that have the best performance, that is, that run as fast as possible and do not slow down the system.

The most important factors that affect the performance of an application are:

- Access to data files within ALL-IN-1
- Access to applications outside ALL-IN-1

When creating an ALL-IN-1 application, keep in mind that good performance often follows good design. An application that makes users move back and forth between forms and requires repeated screen paintings and other nonessential activities, not only frustrates users, but worsens performance. Conversely, the application that allows users to accomplish a task with as few steps as possible is generally the one with the better performance.

Real Versus Perceived Performance

Ensure that the design of your application meets the users' expectations. Often, perceived performance is more important than real performance: an application that performs well can appear slow to users, while an application that performs badly can appear fast.

For example, even though "Working..." messages add to the processing time, they may give the users the impression that the operations in progress are happening quickly.

Similarly, always collect data from the user before beginning processing. If you begin processing before asking for data, users will wonder what is happening. This can mean that you have to save data that could be operated on immediately, but users expect to see a fast response when they first call an application. Once they enter data, they are more prepared to wait a short time.

Use index forms where appropriate to your application. Generally, users are more productive when working from index forms. Although there may be some overhead in first displaying the index form, access to the records thereafter is much quicker.

Using FMS Forms

ALL-IN-1 is heavily form-driven. Even in hardcopy mode, all actions are based on the named data of the current form (or related forms, as in the case of form DEFAULT). Hard copy simply avoids repainting of screens. Therefore, fast access to forms is important for performance. To obtain the maximum performance from forms:

- Use the form libraries supplied by ALL-IN-1.

There is always overhead in opening form libraries. Do not invent new form libraries for new applications, unless they are necessary for controlling access to forms. Using the form libraries supplied by ALL-IN-1 for customizations reduces the overhead of opening form libraries.

- Where possible, keep form sizes below 4000 bytes.

When a form is accessed, FMS allocates workspace. The default workspace size is 4000 bytes. If a form is above this size, more workspace will be allocated, but in segments, usually of 512-byte chunks, thus making the process of allocation longer.

To see the size of the forms in a form library, use the following DCL command:

```
$ FMS/DIR/FULL formlibrary
```

Workspace is not allocated unless the form is displayed. Forms that are accessed for their named data only do not use workspace. Examples of this sort of form are DEFAULT and EM\$INDEX\$OPTIONS. If you have a form that is larger than 4000 bytes, consider putting some of its named data on another form that is not displayed. This is especially useful if you are writing an application in which the same options can be accessed from several forms.

- Always call forms explicitly with the FORM function.

Although a user can access a form directly by entering its name in the choice field of a menu, performance is better if you provide an option in named data that calls the form using the FORM function.

For example, to access form XYZ, the user can enter XYZ in the choice field of any menu. However, if the current menu contains an option XYZ, which simply performs the function FORM XYZ, performance is improved. This is because of the way that ALL-IN-1 processes user input in the choice field of a menu, as described in the chapter on menus in the *Application Programming: Guide*.

If an application requires users to move frequently from a menu (for example, A) to a form (for example, B), include an entry B with the form function to call form B in the named data of menu A. The option calling B need not appear on the menu.

- Limit the number of forms that your application displays.

After a form is displayed, it is held in cache. Until the cache is full, the workspace for a form is held in memory, even when the form is not being used. Thus, if you call a form unnecessarily, the workspace for that form occupies memory until the cache is full.

The number of forms held in cache is determined by a link time constant, called OA\$DSA_FORM_CACHE_SIZE, set in A1LINK.COM. Although it may be changed by editing A1LINK.COM and relinking the ALL-IN-1 image, the optimum value for normal work has been determined to be six forms.

Designing Forms

When designing forms, consider the following guidelines:

- Consider the overhead cost of special screen formatting features.

FMS gives the application designer a variety of special features from which to choose, such as bolding, blinking, reverse video, and drawing boxes.

These features, however, have a cost in terms of the overhead required to do the screen painting. If quick screen painting is desired, avoid these special FMS features

- Avoid a large number of fields in a single form or form set. ALL-IN-1 loads forms field-by-field, not all at once.
- Let users perform an operation without calling a form.

Often, a single operation requires certain information before it can proceed. For example, when selecting a document, the SEL option requires at least a folder name. Normally, you obtain this information by displaying an argument form. However, when users need to enter only a single item, it improves performance if they can enter that item without the overhead of calling another form. In the case of the SEL option, this is achieved by

using the `OA$MENU_REMAINDER` symbol to let the user enter *SEL folder name* in the choice field of a menu. See the named data of the EM menu for an example of how to do this.

Writing Efficient Scripts

When you write a script, bear in mind the following guidelines:

- Use the ALL-IN-1 functions and qualifiers throughout, rather than reverting to Digital Command Language (DCL) or user-written application code. In this way, you avoid leaving the current ALL-IN-1 process or the context of the current field or form.
- Use the following guidelines for `.GOTO` and `.LABEL` script directives.

The first time the script processor meets a `.LABEL` directive, it creates an in-memory table to hold the addresses of labels. It stores any labels it finds as it processes each line. When it comes to a `.GOTO`, it searches the in-memory table and then goes forward through each line of the script until it finds the label. It stores any other labels it finds on the way. However, the in-memory table can hold only 32 entries. Therefore:

- Always start a script with a label to ensure that the in-memory table is created immediately.
 - Never have more than 32 labels in a script.
 - Move backwards with `.GOTO` when possible. A backward `.GOTO` processes faster than a forward `.GOTO`. The further a forward `.GOTO` is from its `.LABEL`, the worse it is.
 - Design the script so that the most common path is the one without branching.
- Do not use the expression `GET OA$FUNCTION = "function"`, where the function will work by itself. `GET OA$FUNCTION` is only necessary to supply a symbol to a function that demands a value. Similarly, avoid the `.FUNCTION`, `.FX`, and `.FZ` directives in a DO script, where they are unnecessary, because they cause `GET` to be performed unnecessarily.
 - Do not fill a form field from a script. Although this is possible, it is much more expensive than filling the field from within the form itself.
 - An alternative to a script is the use of the XOP function to include the script commands within the named data of a form. Use this alternative when the script is simple and small. In general, processing of functions is faster from named data, but be careful not to make forms that are displayed too large.

The XOP function is also useful for pieces of code that are common to several options.

- When checking the value of `OA$STATUS`, use `IFSTATUS` or `IFNOTSTATUS` in preference to the `.IF OA$STATUS EQUAL ...` directive. This also applies to named data.
- Make sure FOR functions stop as soon as possible. You can use `IFSTATUS` to stop a FOR loop as follows:

```
GET OA$STATUS = 1
FOR A DO .IF .x = #x THEN GET OA$STATUS = 0 \IFSTATUS
```

As soon as `.x` begins with `#x`, the loop stops.

- Use the Customization Management (CM) site text library, CMTXL.TXL.

The DO scripts, SCRIPT scripts, and MERGE templates used by your application can be stored in compiled format in the site text library. Accessing the compiled version in the site text library is much faster than accessing the script or template directly. Files for inclusion in the CM site text library are kept in the OA\$SITE_DO_*, OA\$SITE_SCP_*, and OA\$SITE_BLP_* directories. It is the ALL-IN-1 manager's job to ensure that the files in these directories are compiled into the site text library.

Once a file is compiled into the site text library, you call the compiled version by specifying only the name of the file. If you specify a directory containing the file, ALL-IN-1 uses the version in that directory and does not look for a version in the site text library.

- If it is not appropriate to include a file in the site text library, for example, because it is used by only a small number of users, you may want to modify the symbols OA\$TXL_SEARCH_ORDER and OA\$FILE_SEARCH_ORDER. When you call the DO, SCRIPT, or MERGE function, ALL-IN-1 looks for the file in directories specified by these two symbols. These symbols are defined for users within their own individual ALL-IN-1 process, but may be defined dynamically within any application. Modifying the search order for files, by modifying these symbols on a user-by-user basis or application-by-application basis, can result in a significant reduction in file lookups and an improvement in performance. Refer to *ALL-IN-1 Application Programming: Reference Volume 1* for a full description of these symbols.

If your script will not be included in the site text library, do not use long comments within it. You can, however, put comments at the end of the script without any performance penalty.

Using Symbols

When writing scripts, named data, command procedures, and other application software, symbolic data is frequently used. ALL-IN-1 provides a number of ways of referencing this data:

- String literals — “Dog”
- Local symbols that are memory-resident — #DOG
- Permanent symbols that are stored in a Record Management Services (RMS) file — \$DOG
- Field names that reference data on the current form — DOG
- Special symbols that are maintained by ALL-IN-1 and may be readable, writable, or both
- Data set references that point to fields in records of a data set — PROFIL.DIRECT[OA\$USER]
- Symbol substitution — @#CANINE = #DOG = “Dog”

Although all symbol types can be used interchangeably where symbolic data is acceptable to ALL-IN-1, it is often possible that the desired information resides in more than one place (or can be stored in more than one place), and that one place is faster to retrieve from than the other.

- Temporary and permanent symbols compared

Because temporary symbols are memory-resident and permanent symbols are stored in a RMS file, it is more efficient to write scripts, named data, or User-Defined Processes (UDPs) that reference temporary symbols, rather than permanent symbols. Of course, if you want to maintain the value of a symbol from one session to the next, you must store that value in a permanent symbol, since temporary symbols are not maintained from session to session. However, even in this case, it is faster to retrieve the permanent symbol at the beginning of a script, store it in a temporary symbol for the duration of the procedures, and finally store the result in the permanent symbol table.

If too many permanent symbols are created, access time to the permanent symbol table file (.PST) will become slower.

- Special symbols

Many of the ALL-IN-1 special symbols are local representations of data in several of the key ALL-IN-1 data files, such as the user profile or the user's File Cabinet. Rather than use a full data set reference to retrieve the information, you can reference the special symbol, which improves performance. An example is to use OA\$PROFIL_DIRECT, rather than PROFIL.DIRECT[OA\$USER].

Using the Subprocess

When ALL-IN-1 executes a DCL command or command procedure, a subprocess is created. This has a large impact on performance (for example, because of the buffered input/output (I/O) caused by mailbox communications). Therefore, where possible, avoid using the subprocess.

In normal processing, ALL-IN-1 does not use the subprocess. Once a subprocess is created, it remains for the remainder of the session. To avoid using a subprocess:

- Use ALL-IN-1 functions, rather than the equivalent DCL commands, for example, EDIT, PURGE_FILE, RENAME, DELETE_FILE, COPY, QUEUE_BATCH.
- Use the ALL-IN-1 INSTALL and EXECUTE functions to create your own functions. By doing so, you avoid running images from the subprocess.
- If you do not want the subprocess and the main ALL-IN-1 process to communicate, use SPAWN to create a detached process. You can then use the ATTACH command and the SPAWN command to toggle between your main and detached processes.

If you must use the subprocess, use it as little as possible.

Testing Application Performance

The following are suggestions on how to measure the resources used by your application:

- Use the Element Speed Measurement (ESM) option on the CM menu to test keystrokes.
- Use the DECLARE_METER function to place metering hooks in the code and, if you do not consider what you are metering to be a major function, remove the hooks before you move the application to the live area. See the *ALL-IN-1 Application Programming: Reference, Volume 2* for details on the DECLARE_METER function. Also see the *ALL-IN-1 Management Guide* for a description of the metering facility.

APPLICATION PROGRAMMING

- Use the `OA$STAT_GET_STATISTICS` function to get a snapshot of resource usage.
Among the symbols set by this function, the following are useful:
`STAT$CPUTIM` — central processing unit (CPU) time
`STAT$DIRIO` — direct I/Os
`STAT$PAGFLTS` — page faults
`STAT$BUFIO` — buffered I/Os
- Calculate elapsed time using the `OA$DATE_NBS` symbol.
To find the time taken for any part of your application to run, get the value of the symbol `OA$DATE_NBS` immediately before and immediately after your code. The elapsed time is the difference between these two values.
- Use the ALL-IN-1 `OA$LOG_TO_FILE` function to give you the elapsed time and resource usage between keystrokes and functions.
- Use the Trace facility. For example, by using `<OA$TRA_SET IO`, you can see whether a record is fetched directly by key or after a sequential search.

Using Metering

ALL-IN-1 provides the means to collect statistics on resource use of all the major activities by means of metering. The metering collection points also act as DECtrace collection points for customers with DECtrace. The information collected by the meters can be used as a basis for performance analysis and capacity planning.

For consistency, include metering collection points in your code. It is advisable to meter only major functions. For example, use metering for actions such as reading a document or sending a message, but not for validating an address. If you use too many metering collection points, performance of the application suffers and the ALL-IN-1 manager becomes swamped with information.

Programming for Optimal Field Access

This article describes different ways of accessing data that is held in an ALL-IN-1 data set. It also gives hints on how to achieve the fastest data access in your application.

Searching for Records

There are two ways in which ALL-IN-1 searches a data set to find a record: using a sequential search or using an indexed search.

Using a sequential search, ALL-IN-1 examines every record in the data set to find the desired record. With a large data set, a sequential search can be slow.

Using an indexed search, ALL-IN-1 makes use of the key structure of Record Management Services (RMS) files. An RMS file contains an index of the values held in the key field of each record. This index is known as the key of reference, and the entry for an individual record in the key of reference is known as the key to that record. When you access a data set, you specify the key of reference to use and the key value of the record you want to access. ALL-IN-1 searches the key of reference until it finds a key that matches the key you supply. It then accesses the record that corresponds to that key, instead of examining every record. This is known as an indexed search and is usually much faster than a sequential search.

Accessing Data

There are several ways of accessing fields from records in a data set. These are:

- Using a Data Set Reference (DSR)
Use a DSR to access a single field from an individual record.
- Using the %OPEN and %CLOSE special fields in conjunction with a DSR
Use these special fields to access several fields in the same record.
- Using a record selection expression (RSE) with the FOR function
Use an RSE in a FOR loop to select multiple fields from contiguous records.
- Using DATA_FILE subfunctions
Use these subfunctions to access multiple fields from multiple records.

Always use the simplest method that is available for the type of access you require. For example, there is no point in using DATA_FILE subfunctions to access fields in a single record.

Using a DSR

The simplest way to read data from a file is with a DSR. A DSR has the following syntax:

```
dsab.kor.field[key]
```

where:

- *dsab* is the name of the data set access block (DSAB) that accesses the data set.
- *kor* is the name of an RMS key of reference to the data set. This is optional when accessing by primary key.

APPLICATION PROGRAMMING

- *field* is the name of the field that you want to access.
- *key* is an ALL-IN-1 symbol whose value is the RMS key of the record to be accessed.

In the following example, the DSR accesses the DESC field from the FORMAT data set for the record whose key value is WPSPLUS:

```
FORMAT.DESC["WPSPLUS"]
```

Using %OPEN and %CLOSE

If you want to access more than one field from a single record in a data set, use the special fields %OPEN and %CLOSE within your DSR. For example, to access the DESC, DEFAULT_FILE, and DSAB fields from the FORMAT data set for the record whose key value is WPSPLUS:

```
GET FORMAT.%OPEN["WPSPLUS"]
GET .DESC
GET .DEFAULT_FILE
GET .DSAB
GET .%CLOSE
```

Using an RSE

Use RSEs to select multiple fields from contiguous records in a data set. For example, to access the DESC, DEFAULT_FILE, and DSAB fields from the records whose key values begin with ASCII, use an RSE with the FOR function as follows:

```
FOR FORMAT WITH .%KEY BEGINNING "ASCII" DO GET .DESC "," .DEFAULT_FILE "," .DSAB
```

To allow ALL-IN-1 to perform an indexed search, the RSE must specify the key value, or at least the beginning of the key value, and the name of the FMS field that contains the key value. If you do not specify these in your RSE, ALL-IN-1 cannot access the records by key and, therefore, looks through the data set sequentially until it finds a record that matches the search criteria. This makes accessing the data set much slower and results in poorer performance.

When you write an RSE, bear in mind the following rules:

- Use only the EQS, ENS, or BEGINNING relational operators. If you use any other relational operator, ALL-IN-1 performs a sequential search of the whole data set.
- Use the AND operator where possible, instead of the OR operator.
- If you specify an alternate key, specify the name exactly as it is quoted in the file definition language (FDL).
- Reference the FMS field that maps to the first segment of the key by name, rather than using the special field name %KEY.

It is not always possible to follow these rules, but if you do not follow them, you will degrade performance.

ALL-IN-1 sometimes performs a search that is partially indexed and partially sequential. For example, if your RSE uses EQS or BEGINNING when searching for the key field, ALL-IN-1 performs an indexed search. However, if the RSE also contains a Boolean expression that uses another relational operator (such as CONTAINING), ALL-IN-1 performs a sequential search of the records whose key fields match.

To check whether ALL-IN-1 is doing an indexed or a sequential search, use OA\$TRA_SET IO to turn on input/output (I/O) tracing while you are testing an application. The resulting log shows the record accesses, and you can determine whether the application was coded in the optimum manner. Sequential searches show repeated GET_NEXT operations being performed while indexed searches result in LOCATE and GET_BY_KEY operations.

Using DATA_FILE Subfunctions

If you want to access fields from several records in a data set and the are not contiguous, use DATA_FILE subfunctions. The following example accesses the DESC, DEFAULT_FILE, and DSAB fields from the FORMAT data set for the records whose key values are WPSPLUS and ASCII WPS:

```
DATA_FILE OPEN FMT FORMAT
DATA_FILE GET FMT "WPSPLUS" #WPREC
DATA_FILE GET_FIELD FMT DESC #WPDESC
DATA_FILE GET_FIELD FMT DEFAULT_FILE #WPDFT
DATA_FILE GET_FIELD FMT DSAB #WPDSAB
DATA_FILE GET FMT "ASCII WPS" #AWREC
DATA_FILE GET_FIELD FMT DESC #AWDESC
DATA_FILE GET_FIELD FMT DEFAULT_FILE #AWDFT
DATA_FILE GET_FIELD FMT DSAB #AWDSAB
DATA_FILE CLOSE FMT
```

Accessing Data for Field Validation

ALL-IN-1 also searches a data set when you specify validation on a field, using /VALID or /RSE_VALID. Therefore, try to ensure that the search is as efficient as possible.

Whenever possible, use the key field as part of the search. In a /VALID statement, use the key field within the DSAB.FIELD syntax. In a /RSE_VALID statement, specify the key field within the RSE. This applies also in the case of a segmented key, but only to the first segment of the key. When referencing this first segment, ALL-IN-1 does indexed searching; referencing any other segment results in a full sequential search of the data set.

The action performed by each validation qualifier is equivalent to a FOR function as follows:

- /VALID=DSAB.FIELD

equates to:

```
FOR DSAB WITH .FIELD == @OA$FLD_NAME DO OA$VAL_SET_VALID
```

When you use the /VALID qualifier, ALL-IN-1 generates an RSE. In this case, the RSE is DSAB WITH .FIELD == @OA\$FLD_NAME.

If FIELD is the key field, ALL-IN-1 does an indexed search. If it is not the key field, ALL-IN-1 does a sequential search.

- /RSE_VALID = DSAB.FIELD

equates to:

```
FOR DSAB DO .IF .FIELD == @OA$FLD_NAME THEN OA$VAL_SET_VALID
```

When you use the /RSE_VALID qualifier, ALL-IN-1 does not generate an RSE. In this example, no RSE was specified as a parameter. Since ALL-IN-1 does not generate one, it does a sequential search of the whole DSAB. The field name FIELD, appended to the DSAB, is not used as a search criterion.

APPLICATION PROGRAMMING

- /RSE_VALID=DSAB.FIELD WITH .A == A and .B == B

equates to:

```
FOR DSAB WITH .A == A AND .B == B DO IF .FIELD == @OA$FLD_NAME THEN OA$VAL_SET_VALID
```

In this example, ALL-IN-1 will do an indexed search if A or B is the key field or the first segment of the key field. Field FIELD is not part of the RSE, so does not affect the type of search.

SYSTEM MANAGEMENT

Managing an Application for Performance

This article gives guidelines on how to manage applications so that they have the best performance, that is, they run as fast as possible and do not slow down the system.

Managing Form and Text Libraries

- Precompile all form libraries (except DEVELOP.FLB).

There is a large overhead associated with accessing forms that are not in a precompiled form library. Forms are kept in a form library (with the .FLB file type), which is a sequential file. When a noncompiled form library is opened, ALL-IN-1 works with the FMS form driver to build a directory of forms in memory so as to access forms as quickly as possible. The first time a noncompiled form is accessed, the named data of the form is compiled and held in memory. From then on, access to that form is as quick as if the form library was precompiled. However, users have already experienced a delay — first when the form library was opened and again when the form was first accessed. The time taken to build this directory increases exponentially according to the number of forms in the form library. Another disadvantage with a noncompiled form library is that a dynamically compiled form lasts only while the form is in cache. Once it leaves cache, you need to recompile it again.

When you precompile a form library, the named data of the forms in that library is compiled and held in a compiled form library, which has the same name as the .FLB file and has a file type of .FLC. The compiled form library includes a directory of forms: for each form in the compiled form library there is a reference to the address of that form in the noncompiled form library. This ensures that screen data held in the file can be directly accessed without searching through the file.

The development form library (DEVELOP.FLB) is provided as an environment for developing applications. It is not accessible by ordinary users, and its contents are liable to change often. Therefore, you do not need to precompile DEVELOP.FLB.

You compile a form library using the Precompile a form library (PCF) option.

- Install as global sections the compiled form libraries that will be used simultaneously by more than one user.

The cost to the system of installing a form library as a global section in shared memory is small. By installing a form library, users of the forms in the library share the memory used by the library, thus reducing the overall memory requirements of ALL-IN-1. Note, however, that:

- You must not exceed the number of global sections or pages allowed by the system.
- There is little benefit in installing form libraries that are accessed by only one user or are infrequently accessed.
- You must compile the form library before installing it as a global section.
- You compile and, optionally, install a form library using the PCF option.

SYSTEM MANAGEMENT

- Compile and install the text libraries.

DO scripts, SCRIPT scripts, and MERGE templates that are frequently accessed by users should be compiled into a text library (TXL). TXLs are unique to ALL-IN-1. Accessing a file in a TXL provides better performance than accessing the noncompiled version of the file. Once compiled, a TXL should be installed as a global section in shared memory.

You compile and install TXLs from the ALL-IN-1 manager's account. Use the Compile and install CM text library (CCM) option to compile and install the site text library and the Compile and install ALL-IN-1 text library (CTX) option to compile and install the standard text library.

The advantages of compiling and installing a TXL are:

- Scripts and templates are in shared memory. This reduces the memory needs for the system.
- ALL-IN-1 functions are preparsed, so execution is immediate.

Managing Compute-Intensive Applications

ALL-IN-1 is often used as an interface to compute-intensive applications and batch jobs. It can be useful to run interactive and batch jobs simultaneously in this way, but it does pose some capacity and planning problems. These can usually be solved with a little planning and good knowledge of the applications.

Problems can occur with compute-intensive jobs that:

- Use all the central processing unit (CPU) resources and, hence, increase the response time for ALL-IN-1 users
- Are set at too low a priority and have an unacceptable turnaround time

Possible solutions are:

- Dedicate some portion of the available processing power in a VAXcluster system to serving compute-intensive applications. This requires a business decision on whether the applications are important enough to set aside resources for their sole use.
- Group the applications into categories — trivial, low, medium, and high CPU demands, for instance. Trivial jobs can be run interactively, while low, medium, and high compute-intensive jobs are submitted to the appropriate generic queue and, in turn, to queues on all nodes on the VAXcluster system. Each queue can be set up with the appropriate time limits to force compliance to the categories.

Maintaining Files for Better Performance

As well as the following guidelines, consult the VMS manual *Guide to VMS File Applications* for a full discussion of file maintenance.

- Regularly maintain the data files used by applications.

Both ALL-IN-1 files, such as the File Cabinet, and application files will be affected by maintenance. In the case of ALL-IN-1 files, regular maintenance of files can be controlled by the system manager.

For application files, bear the following in mind:

Record Management Services (RMS) indexed files grow dynamically; they do not shrink dynamically. As index levels grow with the increased number of records, access to records becomes slower and it becomes necessary to maintain the data files.

To maintain files, frequently delete unwanted records and reorganize files. Some ALL-IN-1 housekeeping procedures do automatic reorganizations. Use the DCL CONVERT command for other files. Refer to the documentation on the VMS Convert and Convert/Reclaim Utility.

- Consider the optimum size of buckets when designing an application.

A large data bucket size is better when records are accessed close together on the file. For example, the default data bucket size for DOCDB.DAT is 12. This is because the common user operations that require access to DOCDB, such as Index a folder or Read new mail, often entail reading records that are in the same area on the file as each other or records just accessed.

However, for files where access to records is random, a data bucket size of 2 or 3 is enough.

The index bucket size need not be as large as the data bucket size. For examples of data and index buckets, look at the data files that ALL-IN-1 uses. You can find examples of typical data files in the directories OA\$DATA_SHARE and OA\$DATA_LLIV. Most data files have an equivalent file definition language (FDL) file, for example, ACITEM.DAT and ACITEM.FDL. Use ANALYZE/RMS_FILE/STATISTICS at Digital Command Language (DCL) command level to examine the internal structure of these files or use the many examples of FDL files in OA\$LIB_SHARE as templates for creating your own files.

- Put global buffers on a file if it will be accessed simultaneously by more than one user. Global buffers are more effective when a file is read more than it is written to. A general rule is to have enough global buffers to hold all the index buckets and at least one data bucket.

Problem in X.400 Addresses with Multiple Organizational Units

A problem was discovered when sending messages from or through either of the following two sets of products by way of the Message Router X.400 Gateway (MRX) to an X.400 recipient whose Originator/Recipient (O/R) name includes more than one organizational unit:

1. ALL-IN-1 Electronic Messaging, ALL-IN-1 MAIL
2. Message Router/P Gateway (MRP), Message Router/S Gateway (MRS), Message Router Telex Gateway (MR-Telex), ULTRIX Mail Connection (UMC)

The problem occurs when the Message Router Directory Service generates the address of the recipient. The messages from one of the sets of products will be rejected by the receiving Message Transfer Agent (MTA). This problem may not affect you now; however, if you introduce one of the other set of products to your mail system or start sending messages to an MTA that is sensitive to the order of organizational units in O/R names, you will experience the problem of messages not being delivered.

If this problem is likely to affect your mail network, please contact your local Customer Support Center (CSC).

For more information about MRX and O/R names, refer to the Message Router X.400 Gateway documentation.

This problem arises because of differences between the two sets of products. The first set does the following:

- ALL-IN-1 Electronic Messaging

ALL-IN-1 uses the Directory Service to supply an X.400 address only when users specifically search the Directory Service database (known in ALL-IN-1 as the Mail Directory) for the address using the Search mail directory (SMD) option or when ALL-IN-1 is configured to use the Directory Service database primarily.

Users can store the returned address as a nickname to use later. Any addresses obtained and stored in this way are still classed as being supplied by the Directory Service.

- ALL-IN-1 MAIL

ALL-IN-1 MAIL uses the Directory Service to supply an X.400 address only when users specifically search the Directory Service database for the address.

Users can store the returned address in their personal address books to use later. Any addresses obtained in this way and stored in a personal address book are still classed as being supplied by the Directory Service.

The second set of products does this differently, in the following way:

- Message Router/P Gateway

MRP always uses the Directory Service to supply the address of an X.400 recipient.

- Message Router/S Gateway

MRS always uses the Directory Service to supply the address of an X.400 recipient.

- Message Router Telex Gateway

MR-Telex uses the Directory Service to supply the address of an X.400 recipient if it is configured to do so.

SYSTEM MANAGEMENT

- **ULTRIX Mail Connection**

UMC users employ the Directory Service to supply an X.400 address only when they use the address lookup utility that is supplied with UMC.

Users can store the returned address in the alias file to use later. Any addresses obtained and stored in this way are classed as being supplied by the Directory Service.

The two sets of products encode the recipient's O/R name in different ways in the National Bureau of Standards (NBS) form of the message. When the MRX receives the NBS form of the message, it translates the message to its X.409 form. During this translation, MRX handles the two different encodings of the O/R name in different ways. Hence, the organizational units in the recipient's O/R name in the X.409 form of messages sent from the two product sets are in opposite orders of significance. This difference in the order of significance of the organizational units in the recipient's O/R name causes a problem if all three of the following conditions exist:

- You send messages, using products from both sets, to X.400 recipients whose O/R names include multiple organizational units.
- The addresses are supplied by the Directory Service (specifically, the MHS User Identifier field in the Directory Service entry).
- The receiving MTA is sensitive to the order of organizational units in O/R names.

The messages from one of the sets of products will be rejected by the receiving MTA.

The recommended way of setting up Directory Service entries for recipients whose O/R names contain multiple organizational units is to order the organizational units in ascending order of significance in the MHS User Identifier field, that is, from least to most significant. For example, you would use the following MRXMAN command:

```
BUILD SUBSCRIBER /SURNAME=jones
/GIVENNAME=david
/PCOUNTRY=gb
/ADMD=mailnet
/PRMD=bytecorp
/ORGNAME=eurosales
/MHSADDRPART=nodeg
/MHSMAILBOX=b
/MHSUSERID="david jones@cou=gb@admin=mailnet@private
=bytecorp@organization=eurosales@unit=pcsales@unit
=london@unit=southeast@unit=uksales"
```

where:

- *uksales* is the most significant organizational unit.
- *pcsales* is the least significant.
- *unit* is the keyword used to denote the organizational unit.

This gives the following Directory Service entry:

```

ADMDNAME   (Admin Domain)       : MAILNET
PCOUNTRY   (Country Code)       : GB
GIVENNAME  ( First Name(s) )    : DAVID
MRREVLOOKUP (MR Address)        : david jones@cou=gb@admin=mailnet@private
                                     =bytecorp@organization=eurosales@unit=pc
                                     sales@unit=london@unit=southeast@unit=uk
                                     sales@B@NODEG
NAME        (Common Name)       : DAVID JONES
ORGNAME     (Organization Name)  : EUROSALLES
PRMDNAME    (Private Domain Name): BYTECORP
SURNAME     : JONES
MASTER COPY OWNING NODE       : ADDLE
++ ORADDRESS 1
MHSADDRPART (MHS Address)       : NODEG
MHSMAILBOX  (MHS Mailbox)       : B
MHSUSERID   (MHS User Id)       : david jones@cou=gb@admin=mailnet@private
                                     =bytecorp@organization=eurosales@unit=pc
                                     sales@unit=london@unit=southeast@unit=uk
                                     sales

```

Cumulative Index

The index on the following pages contains entries from the August and November 1990 and February 1991 issues of the *ALL-IN-1 Information Update*, as well as from this issue. For each entry, the issue date and page number are shown.

INDEX

8-bit characters, *August 1990*, 11
in user names, *August 1990*, 11

A

A1PAT509, *November 1990*, 5
A1PAT510, *November 1990*, 7
A1PAT511, *November 1990*, 8
A1PAT512FOREIGN, *February 1991*, 5
A1TRACE.LOG, *November 1990*, 13
 contents of, *November 1990*, 14
 viewing, *November 1990*, 14
Access violation, *November 1990*, 5, 6, 8, 9
Addresses, long
 not matching full names, *November 1990*, 6
 reading messages with, *November 1990*, 8
 showing message status, *November 1990*, 8
Address validation, *November 1990*, 6
Alias file, *May 1991*, 20
ALL-IN-1 subprocess, *November 1990*, 23
AND operator
 See Operators
Applications
 compute-intensive
 managing, *May 1991*, 16
 problems, *May 1991*, 16
 solutions to problems, *May 1991*, 16
ASCII documents
 reading, *November 1990*, 7
ATTACH function
 See Functions
Auto forward, *November 1990*, 5

B

Balance mail areas, *November 1990*, 5
Batch
 VAXcluster queues, *August 1990*, 16
BEGINNING operator
 See Operators
BIND function, *November 1990*, 19
BINDW function, *November 1990*, 19
Boolean expression, *May 1991*, 12
Break, *August 1990*, 43
Broadcast handling, *November 1990*, 7

C

CAL\$CHECK_DATE DSAB, *August 1990*, 65
CALLBACK, *November 1990*, 7
CDA support, *November 1990*, 11
CDA_CONVERT function, *February 1991*, 21
Common-environment VAXcluster system
 running ALL-IN-1, *August 1990*, 15
Communications Control, *August 1990*, 43
Compile and install CM text library (CCM)
 menu option
 See Menu options
Compound documents (CDA)
 defined, *February 1991*, 19
 storage format, *February 1991*, 19
 STORED SEMANTICS attribute, *February 1991*, 19
 support in Version 2.4, *August 1990*, 3
Compute-intensive applications
 See Applications
CONTAINING operator
 See Operators
COPY function
 See Functions
CTRL/T, *November 1990*, 23
CTRL/Y, *November 1990*, 23
Customization Management, *August 1990*, 45
 extending, *August 1990*, 45
Customizations
 after installing patches, *November 1990*, 7
CXP subfunctions, *August 1990*, 43

D

Data
 accessing, *May 1991*, 11
Data buckets, *May 1991*, 17
 optimum size, *May 1991*, 17
Data sets, phantom, *November 1990*, 19
DATATRIEVE, *August 1990*, 36
DATA_FILE subfunctions
 See Subfunctions
Date format
 user's preferred, *August 1990*, 65
Date validation, *August 1990*, 65
 form field, *August 1990*, 65

DATE_CONVERT function, *August 1990*, 65
 DDIF, *February 1991*, 19
 DECLARE_METER function
 See Functions
 DEClaser features, *February 1991*, 13
 DEClaser printer
 Adding support for, *February 1991*, 9
 Duplex attributes, *February 1991*, 11
 Manual feed option attribute, *February 1991*, 13
 Tray selection attributes, *February 1991*, 13
 Using, *February 1991*, 9
 DECnet, *August 1990*, 24, 43
 DECtrace, *May 1991*, 10
 Deferred messages, *August 1990*, 23
 retry count, *November 1990*, 8
 Define Communications Lines, *August 1990*, 44
 Define Computer Systems, *August 1990*, 44
 DELETE_FILE function
 See Functions
 Development area, *August 1990*, 45
 Distribution lists, *November 1990*, 6
 DOWN ARROW, *November 1990*, 6
 Document formatting queue
 in a VAXcluster system, *August 1990*, 17
 Documents
 compound, *February 1991*, 19
 selecting, *August 1990*, 13
 Document transfers, *November 1990*, 7
 DO function
 See Functions
 DO scripts, *August 1990*, 61; *May 1991*, 7, 8, 16
 See also Scripts
 DOTS, *February 1991*, 19
 DSAB (data set access block), *May 1991*, 13
 DSR (Data Set Reference), *May 1991*, 11
 using, *May 1991*, 11
 DTIF, *February 1991*, 19

E

EDIT function
 See Functions
 Elapsed time
 calculating, *May 1991*, 10
 Electronic Messaging, *November 1990*, 5
 address validation, *November 1990*, 6
 deferred messages, *November 1990*, 8
 FIND, *November 1990*, 6
 long addresses, *November 1990*, 6, 8
 multiple delivery, *November 1990*, 8
 nondelivery, *November 1990*, 8
 paper mail, *November 1990*, 8
 sent date, *November 1990*, 8

Electronic Messaging (Cont.)
 ULTRIX mail gateway, *November 1990*, 6
 Empty Wastebasket failure, *November 1990*, 8
 ENS operator
 See Operators
 EQS operator
 See Operators
 Escape sequence, *August 1990*, 43
 ESM (Element Speed Measurement) menu
 option, *May 1991*, 9
 EXECUTE function
 See Functions
 External reference, *February 1991*, 19

F

Fetcher, *August 1990*, 24; *November 1990*, 7
 Field validation, *May 1991*, 13
 Files
 backing up, *February 1991*, 23
 File definition language (FDL), *May 1991*, 17
 Fixes
 issuing, *February 1991*, 3
 FMS
 formatting features
 examples, *May 1991*, 6
 overhead costs, *May 1991*, 6
 FMS form driver, *May 1991*, 15
 FMS forms, *August 1990*, 57
 FOREIGN DSAB, *August 1990*, 63
 FOR function
 See Functions
 Format
 of compound documents, *February 1991*, 19
 converting, *February 1991*, 21
 Formatter queue
 in a VAXcluster system, *August 1990*, 17
 Form directory, *May 1991*, 15
 Form driver
 FMS, *May 1991*, 15
 FORM function
 See Functions
 Form library, *August 1990*, 49; *May 1991*, 15
 compiling, *May 1991*, 3
 development (DEVELOP.FLB), *May 1991*, 15
 installing in a VAXcluster system, *August 1990*, 19
 Forms
 compiling
 with unresolved special symbols, *May 1991*, 3
 designing, *May 1991*, 6
 FMS, *May 1991*, 5
 FORM function, *May 1991*, 6
 libraries, *May 1991*, 5, 15
 number limit, *May 1991*, 6

Forms (Cont.)

optimum value, *May 1991*, 6
performing an operation without calling,
May 1991, 6

size

examples, *May 1991*, 5
with .MORE directive, *May 1991*, 3

Forms compiler

reporting an error, *May 1991*, 3

Functions

ATTACH, *May 1991*, 9
BIND, *November 1990*, 19
BINDW, *November 1990*, 19
CDA_CONVERT, *February 1991*, 21
COPY, *May 1991*, 9
DATE_CONVERT, *August 1990*, 65
DECLARE_METER, *May 1991*, 9
DELETE_FILE, *May 1991*, 9
DO, *May 1991*, 8
EDIT, *May 1991*, 9
EXECUTE, *May 1991*, 9
FOR, *May 1991*, 7
FORM, *May 1991*, 6
GET, *May 1991*, 7
GET_BY_KEY, *May 1991*, 13
GET_NEXT, *May 1991*, 13
GET_RMS_SEMANTIC_TAG, *February 1991*, 20
INSTALL, *May 1991*, 9
LOCATE, *May 1991*, 13
MERGE, *May 1991*, 8, 16
OA\$LOG_TO_FILE, *May 1991*, 10
OA\$STAT_GET_STATISTICS, *May 1991*, 10
PURGE_FILE, *May 1991*, 9
QUEUE_BATCH, *May 1991*, 9
RENAME, *May 1991*, 9
SCRIPT, *May 1991*, 8
SPAWN, *May 1991*, 9
XOP, *May 1991*, 7
.FUNCTION script directive, *February 1991*, 15
.FX script directive, *February 1991*, 15

G

GET function

See Functions

GET_BY_KEY function

See Functions

GET_NEXT function

See Functions

GET_RMS_SEMANTIC_TAG function,

February 1991, 20

Global buffers, *May 1991*, 17

GOLD MESSAGE, *November 1990*, 7

GOLD TRACE, *November 1990*, 14

H

ALL-IN-1 image

privileges needed, *August 1990*, 9

Help, *August 1990*, 55

customizing, *August 1990*, 55

keysize, *August 1990*, 55

keyword, *August 1990*, 55

library, *August 1990*, 55

Heterogeneous VAXcluster system

See Multiple-environment VAXcluster system

HLPSOURCE.HLB, *August 1990*, 56

Homogeneous VAXcluster system

See Common-environment VAXcluster system

I

Image file

installing in a VAXcluster system, *August 1990*, 18

INCREMENT_METER function, *November 1990*, 7

Indexed search, *May 1991*, 13

for records, *May 1991*, 11

Indexes

search phrases, *November 1990*, 8

Index forms

using phantom data sets, *November 1990*, 19

INSTALL function, *May 1991*, 9

See also Functions

with CALLBACK, *November 1990*, 7

Installing form library

in a VAXcluster system, *August 1990*, 19

Installing image file

in a VAXcluster system, *August 1990*, 18

Installing text library

in a VAXcluster system, *August 1990*, 19

IVP (Installation Verification Procedure), *May 1991*, 3

K

Key of reference, *May 1991*, 11

L

LIB\$FREE_VM error, *November 1990*, 8

Live area, *August 1990*, 45

Live link, *February 1991*, 19

LOCATE function

See Functions

Lock element (LE), *August 1990*, 52

Long addresses

not matching full names, *November 1990*, 6
reading messages with, *November 1990*, 8

Long addresses (Cont.)

showing message status, *November 1990*, 8

M

Mail directory, *August 1990*, 21

Mail Directory, *November 1990*, 6

Meeting notices, *November 1990*, 7

Menu options

Compile and install CM text library (CCM),
May 1991, 16

Precompile a form library (PFC), *May 1991*,
15

Search mail directory (SMD), *May 1991*, 19

MERGE function

See Functions

Message

priority for, *August 1990*, 35

Message Router Directory Service, *May 1991*,
19

setting up entries, *May 1991*, 20

Metering

using, *May 1991*, 10

Modem

autodial, *August 1990*, 43

Move to live area (MLA), *August 1990*, 46

MRP (Message Router/P Gateway), *May 1991*,
19

MRS (Message Router/S Gateway), *May 1991*,
19

MR-Telex (Message Router Telex Gateway),
May 1991, 19

MTA (Message Transfer Agent), *May 1991*, 20

Multiple-environment VAXcluster system
running ALL-IN-1, *August 1990*, 15

N

Named data, *May 1991*, 7

EM menu, *May 1991*, 6

NETWORK.DAT, *February 1991*, 23

Nonstandard files, *August 1990*, 63

O

OA\$DAF_y.DAT, *February 1991*, 23

OA\$DATA:NETWORK.DAT, *February 1991*,
23

OA\$DATA:PENDING.DAT, *February 1991*, 23

OA\$DATA:PROFILE.DAT, *February 1991*, 23

OA\$DATA_LLV, *May 1991*, 17

OA\$DATA_SHARE, *May 1991*, 17

OA\$DATA_SHARE:CM_SDC_ELEMENTS.DAT,
November 1990, 11

OA\$FILE_SEARCH_ORDER, *August 1990*,
50; *May 1991*, 8

OA\$LIB_SHARE, *May 1991*, 17

OA\$MTI_ERR, *August 1990*, 35, 36

OA\$MTI_LOG, *August 1990*, 35

OA\$SHARY:OA\$DAF_y.DAT, *February 1991*,
23

OA\$STATUS

checking value, *May 1991*, 7

OA\$TABLE_DSAB, *November 1990*, 25

specifying numbers in, *November 1990*, 25

using hyphens in, *November 1990*, 25

OA\$TEXT_DSAB, *August 1990*, 63

OA\$TRA_SET IO, *May 1991*, 13

OA\$TXL_SEARCH_LAST symbol, *August*
1990, 61

OA\$TXL_SEARCH_ORDER symbol, *August*
1990, 61; *May 1991*, 8

OAHELP.HLB, *August 1990*, 56

OAMTISEND.COM, *August 1990*, 26

Operators

AND, *May 1991*, 12

BEGINNING, *May 1991*, 12

CONTAINING, *May 1991*, 12

ENS, *May 1991*, 12

EQS, *May 1991*, 12

P

Paper mail

formatting, *November 1990*, 8

Patches, *November 1990*, 5

A1PAT509, *November 1990*, 5

A1PAT510, *November 1990*, 7

A1PAT511, *November 1990*, 8

A1PAT512FOREIGN, *February 1991*, 5

available for ALL-IN-1 Version 2.3,
February 1991, 5

customizations after installing, *November*
1990, 7

process for Version 2.3, *February 1991*, 4

process for Version 2.4, *February 1991*, 3

SUS, *February 1991*, 3

PENDING.DAT, *February 1991*, 23

Performance

cost, *May 1991*, 5

designing an application for, *May 1991*, 5

maintaining files for, *May 1991*, 16

managing an application for, *May 1991*, 15

perceived, *May 1991*, 5

programming an application for, *May 1991*,
5

speed of, *May 1991*, 15

testing applications for, *May 1991*, 9

Permanent symbol table file (.PST), *May 1991*,
9

Phantom data sets, *November 1990*, 19

Phone, *August 1990*, 43

Precompile a form library (PCF) menu option

See Menu options

Privileges, *August 1990*, 9

Profile

Profile (Cont.)
 large entry in Department field, *November 1990, 5*
 PROFILE.DAT, *February 1991, 23*
 Programming
 for optimal field access, *May 1991, 11*
 Protect an element from changes (PFC),
 August 1990, 52
 PURGE_FILE function
 See Functions

Q

Qualifiers
 /RSE_VALID, *May 1991, 13*
 /VALID, *May 1991, 13*
 QUEUE_BATCH function
 See Functions

R

Reading ASCII documents, *November 1990, 7*
 Receive a document from VMS (RV), *August 1990, 63*
 Receive without conversion (RVC), *August 1990, 63*
 Recognition
 using OA\$TABLE, *November 1990, 25*
 Record session (RS), *August 1990, 43*
 RENAME function
 See Functions
 Retirement of support
 Version 2.2, *November 1990, 3*
 Retry count
 deferred messages, *November 1990, 8*
 RMS, *May 1991, 11*
 checking data files for errors, *February 1991, 23*
 RSE (record selection expression)
 rules, *May 1991, 12*
 using, *May 1991, 12*
 with the FOR function, *May 1991, 11*
 writing, *May 1991, 12*

S

Script directives
 .GOTO, *May 1991, 7*
 .LABEL, *May 1991, 7*
 SCRIPT function
 See Functions
 Scripts
 DO mode, *February 1991, 15; May 1991, 7, 16*
 nesting, *February 1991, 17*
 SCRIPT, *May 1991, 16*
 writing guidelines, *May 1991, 7*
 SCRIPT scripts, *August 1990, 57, 61; May 1991, 8, 16*

SCRIPTS Script, *February 1991, 15*
 SDAF
 See shared document attributes file,
 February 1991, 23
 Search mail directory (SMD)
 See Menu options
 Search order, *August 1990, 62*
 Search phrases
 with multilingual characters, *November 1990, 8*
 Semantic tag
 defined, *February 1991, 19*
 SENDCHECK.EXE, *August 1990, 26*
 Sender, *August 1990, 23*
 batch job, *August 1990, 35*
 log, *August 1990, 23*
 message processing time, *August 1990, 28*
 queue, *August 1990, 23*
 run time, *August 1990, 28*
 tuning, *August 1990, 23*
 Sent date on messages, *November 1990, 8*
 Sequential search, *May 1991, 13*
 for records, *May 1991, 11*
 Set date option, *November 1990, 6*
 Set Working Conditions (SWC)
 date format, *August 1990, 65*
 Shared document attributes file (SDAF)
 checking for RMS errors, *February 1991, 23*
 reorganizing, *February 1991, 23*
 SPAWN, *November 1990, 23*
 See also Functions
 Special fields
 %CLOSE, *May 1991, 11, 12*
 %KEY, *May 1991, 12*
 %OPEN, *May 1991, 11, 12*
 Special symbols, *May 1991, 9*
 unresolved, *May 1991, 3*
 Starting ALL-IN-1
 in a VAXcluster system, *August 1990, 18*
 Stopping ALL-IN-1
 in a VAXcluster system, *August 1990, 18*
 STORED SEMANTICS attribute, *February 1991, 19*
 Subfunctions
 DATA_FILE, *May 1991, 11, 13*
 Subprocess
 avoiding use of, *May 1991, 9*
 Subprocess, ALL-IN-1, *November 1990, 23*
 Support Update Software (SUS), *February 1991, 3*
 Switch log files (SL), *August 1990, 36*
 Symbols
 comparing, *May 1991, 9*
 OA\$DATE_NBS, *May 1991, 10*
 OA\$DA_FORM_CACHE_SIZE, *May 1991, 6*

Symbols (Cont.)

OA\$FILE_SEARCH_ORDER, *May 1991*, 8
OA\$MENU_REMAINDER, *May 1991*, 6
OA\$PROFIL_DIRECT, *May 1991*, 9
OA\$TXL_SEARCH_LAST, *August 1990*, 61
OA\$TXL_SEARCH_ORDER, *August 1990*,
61; *May 1991*, 8
using, *May 1991*, 8
SYS\$CLUSTER_NODE, *August 1990*, 41

T

Templates, *August 1990*, 61
Terminal session disconnection, *November 1990*, 8
Text elements, *August 1990*, 50
Text library, *August 1990*, 50, 61; *May 1991*, 15
CM, *August 1990*, 61; *May 1991*, 8
location of files, *May 1991*, 8
compiling, *May 1991*, 16
installing, *May 1991*, 16
installing in a VAXcluster system, *August 1990*, 19
managing, *May 1991*, 15
noncompiled, *May 1991*, 15
precompiled, *May 1991*, 15
source locations, *August 1990*, 51
Time Management, *November 1990*, 6
meeting notices, *November 1990*, 7
Trace facility, *November 1990*, 13
to create UDPs, *November 1990*, 13
turning off, *November 1990*, 14
turning on, *November 1990*, 14
use to trace records, *May 1991*, 10
Trace log file
listing without exiting, *May 1991*, 3
Tracing
See Trace facility
Training
modifying, *August 1990*, 57
Tuning ALL-IN-1
in a VAXcluster system, *August 1990*, 16
TXL, *August 1990*, 50, 61

U

UAF (user authorization file), *August 1990*, 22
UDP
See User-defined processes
ULTRIX Mail Connection (UMC), *May 1991*, 20
ULTRIX mail gateway, *November 1990*, 6
UMC, *November 1990*, 6
User accounts
creating, *August 1990*, 21
User-defined processes(UDPs), *November 1990*, 13 to 17

User-defined processes(UDPs) (Cont.)

example of, *November 1990*, 15
script directives in, *November 1990*, 17
UDP names in, *November 1990*, 13
using UDPs to automatically create,
November 1990, 16
writing, *November 1990*, 13
User names with apostrophes, *November 1990*, 5

V

Validation
date, *August 1990*, 65
using OA\$TABLE, *November 1990*, 25
VAXcluster system
alias, *August 1990*, 16
batch queues, *August 1990*, 16
common-environment
running ALL-IN-1, *August 1990*, 15
formatter queue, *August 1990*, 17
installing form library, *August 1990*, 19
installing image file, *August 1990*, 18
installing text library, *August 1990*, 19
multiple-environment
running ALL-IN-1, *August 1990*, 15
starting ALL-IN-1, *August 1990*, 18
stopping ALL-IN-1, *August 1990*, 18
tasks, *August 1990*, 16
tuning, *August 1990*, 16
Version 2.2
retirement of support, *November 1990*, 3
Version 2.3 patch process, *February 1991*, 4
Version 2.4 patch process, *February 1991*, 3
VMS logical names, *August 1990*, 41
VMSmail import, *November 1990*, 7

X

X.400
problem in addresses with multiple
organizational units, *May 1991*, 19
XOP function
See Functions

Attention U.S. Software Contract Customers Submitting Software Problems

As part of Digital's continuing effort to improve customer satisfaction, U.S. Software Product Services (SPS) has changed the process for U.S. customers to submit software problems. These customers should report their software problems by calling their Customer Support Center (CSC) or by accessing either the Digital Software Information Network (DSIN) or DSNlink.

The CSCs have the technical expertise to quickly respond to many of our customers' problems without requiring Engineering intervention. Because many software problems are duplicates of problems that have already been addressed by Engineering, the CSCs can immediately respond to many inquiries.

If you do not have access to a CSC, submit your software problem to:

**Digital Equipment Corporation
P.O. Box 1167
Alpharetta, Georgia 30239-1167**

This address replaces the one currently found on the instruction sheet of the Software Performance Report (SPR) form.

Digital Equipment Computer Users Society

Benefits of Belonging

The Digital Equipment Computer Users Society (DECUS) is one of the largest and most respected users groups in the computer industry today. Membership in DECUS, which is free and voluntary, affords the individual user information and services not found anywhere else.

DECUS provides an environment where users of Digital Equipment Corporation products can share information with other users and with Digital. Members can find out the latest news on Digital's hardware, software, and educational products. The feedback exchange with Digital allows the users of Digital's products a voice in the direction the company takes in the future.

Founded in 1961, DECUS has three autonomous chapters worldwide—DECUS U.S., DECUS Europe, and DECUS General International Area (GIA). DECUS services and activities are shared between these chapters through mutual agreements.

All DECUS services promote the exchange of information in a commercial-free environment. These services include:

Special Interest Groups (SIGs)

These groups, formed around an area of common interest, exist for a variety of hardware, operating systems, languages, applications, and marketing areas. Participation in these groups allows contact among fellow users to exchange information and share technical expertise in the area of most interest to the user.

Local Users Groups (LUGs) and National Users Groups (NUGs)

LUGs and NUGs are licensed groups of individuals who gather to share information with other users on a periodic basis. Not only do these people share professional interest, but they also share geographic and cultural ties. Often, Digital representatives attend these meetings to discuss new products and services and supply updates on existing policies and procedures.

Symposia

DECUS holds symposia each year in the different chapters, two per year in the U.S. These meetings provide unique opportunities for users with a wide spectrum of experience to meet for up to five intensive days of technical exchange. Included in symposia activities are workshops, clinics, panels, tutorials, and formal paper presentations. Digital participates in symposia by sending product group managers and developers to discuss strategies, products, problems, and solutions.

Communications

The flow of information among users, as well as between users and Digital, is the primary goal of DECUS. Various publications are published by DECUS to support this communication. They include chapter newsletters and *The Proceedings*, a technical volume published after each symposium.

DECUS also publishes the Special Interest Groups *SIGs Newsletter* on a monthly basis. The newsletter is divided into Special Interest Group sections and provides specialized information pertaining to specific Digital products. The monthly newsletter and *The Proceedings* are available through the DECUS Subscription Service.

Seminars

DECUS recognizes the need for ongoing training opportunities that help members keep pace with the changing technologies of the computer industry. To that end, DECUS sponsors 1-day technical seminars throughout the year in various locations. These seminars are developed and presented by DECUS members who are experienced industry professionals.

Program Library

The DECUS Program Library is the main vehicle for the exchange of public domain software among users of all Digital systems. The library contains over 1000 software programs written and voluntarily submitted by users. These programs include compilers, editors, utilities, numerical and statistical functions, as well as games and graphic routines. The library publishes a software catalog on an annual basis that lists and describes all the DECUS programs.

DECUServe

DECUServe is an electronic conferencing system based on VAX Notes. The network is available to all U.S. members 24 hours a day, 7 days a week for an annual subscription fee.

Joining DECUS

You are cordially invited to join with over 100,000 other users of Digital products around the world and begin to share your experiences, both the successes and problems.

For more information, contact the appropriate DECUS Chapter Office.

DECUS Chapter Offices—Worldwide

DECUS U.S.

DECUS U.S.
333 South Street, (SHR1-4/D30)
Shrewsbury, Massachusetts 01545-4112
Tel. (508) 841-3389

DECUS Europe

DECUS Europe Headquarters
12 Avenue des Morgines
Case Postale 176
CH-1213 Petit-Lancy 1
Switzerland

DECUS BELUX
Luchtschipstraat 1
1, Rue de l'Aeronef
B-1140 Evere (Brussels)
Belgium

DECUS Finland
P.O. Box 6
SF-02201 Espoo
Finland

DECUS Holland
Europalaan 44
P.O. Box 9212
3506 GE Utrecht
The Netherlands

DECUS Ireland
c/o Digital Equipment Ireland Ltd.
Park House, North Circular Road
IRL-Dublin 7
Ireland

DECUS Italia
Viale Monza 338
I-20128 Milano
Italy

DECUS At-Large Chapter
12 Avenue des Morgines
Case Postale 176
CH-1213 Petit-Lancy 1/GE
Switzerland

DECUS Denmark
c/o Digital Equipment Corporation A/S
Aadalsvej 99
DK-2970 Hoersholm
Denmark

DECUS France
Parc du Bois Briard
9/13 avenue du Lac, BP 235
F-91007 Evry Cedex
France

DECUS Iceland
c/o DECUS Denmark

DECUS Israel
c/o Digital Equipment
Acadia Junction
Herzlia 46 733
Israel

DECUS Muenchen
Freischuetzstrasse 91, Postfach 810247
D-8000 Muenchen 81
Federal Republic of Germany

DECUS Chapter Offices—Worldwide (Continued)

DECUS Norway
Ammerudveien 22
N-0958 Oslo 9
Norway

DECUS Portugal
c/o Digital Equipment Portugal Lda.
Empreendimento Torres das Amoreiras
Av. Eng. Duarte Pacheco, Torre 1-9 andar
P-1000 Lisbon
Portugal

DECUS Spain
c/o Digital Equipment Corporation SA
Cerro del Castanar 72, Mirasierra
E-28034 Madrid
Spain

DECUS Sweden
Box 5194
S-402 26 Gothenburg
Sweden

DECUS Switzerland
DECpark
Uberlandstrasse 1 Postfach
CH-8600 Dubendorf 1
Switzerland

DECUS U.K., Ireland, and Middle East
Queen's House
Forbury Road
GB-Reading, RG 1 3JJ
U.K.

DECUS GIA (General International Area)

DECUS GIA Headquarters
100 Nagog Park AKO1-1/B11
Acton, Massachusetts 01720
U.S.A.

DECUS South Pacific Chapter
Australia Office
Locked Bag 16, 410 Concord Road
Rhodes, New South Wales 2138
Australia

DECUS
New Zealand Office
P.O. Box 8610
Symonds Street
Auckland 3, New Zealand

DECUS Canada
505 University Avenue, 15th Floor
Toronto, Ontario M5G 1X4
Canada

DECUS Far East
Digital Equipment Corporation
19-21 Fleet House
38 Gloucester Road
Wanchai, Hong Kong

DECUS Japan
Nihon Digital Equipment KK
Sunshine 60, P.O. Box 1135
1-1. Higashi Ikebukuro 3 - chome
Toshima-ku Tokyo
Japan 170

DECUS South America
Digital Equipment do Brasil Ltda.
Av. Presidente Wilson
231 26th Floor
20030 Rio de Janeiro, RJ
Brasil

DECUS Latin America/Carribean Chapter
DECUS LACC
800 Fairway Drive
Suite 400
Deerfield Beach, FL 33441
U.S.A.

DECUS SUBSCRIPTION SERVICE
SIGs NEWSLETTER
U.S. Chapter Proceedings
(U.S. Chapter Members Only)

As a member of the DECUS U.S. Chapter, you are entitled to contribute and subscribe to the DECUS monthly publication, *SIGs Newsletter*. You also have the opportunity to subscribe to the Symposia Proceedings which are a compendium of the reports from various speakers at the U.S. National DECUS Symposia.

- The order form below must be used as an invoice
- All checks must be made payable to DECUS
- All orders **must** be paid in full
- \$25.00 minimum for credit card orders
- No refunds will be made
- The address provided below will be used for all DECUS mailings, e.g., membership, subscription service, and symposia
- SIGs Newsletter price is for a one-year subscription beginning the month following receipt of payment

.....

Name _____ DECUS Member No. _____

Company _____

Address _____

City _____ State _____ Zip _____

Phone _____

Subscription Service Offering	Qty	Unit Price	Total
SIGs Newsletter	_____	\$40.00	_____
Spring '90 Proceedings (SP0)	_____	15.00	_____
Fall '90 Proceedings (FA0)	_____	15.00	_____
Spring '91 Proceedings (SP1)	_____	15.00	_____
Fall '91 Proceedings (FA1)	_____	15.00	_____
Total Cost of Subscription			\$ _____

☐ MasterCard® ☐ VISA® ☐ DINERS CLUB® ☐ CARTE BLANCHE® ☐ American Express®

Card No. _____ Exp. Date _____

I understand that there will be no refunds, even if I decide to cancel my subscription.

Signature _____

For Digital Employee Use Only

Badge No. _____ CC: _____

CC Mgr Name _____

CC Mgr Signature _____

For DECUS Office Only

Check No. _____

Bank No. _____

Amount \$ _____

Send to: Subscription Service, DECUS, 333 South Street (SHR1-4/D30), Shrewsbury, MA 01545-4112, (508) 841-3389.

Note: DECUS Europe and DECUS GIA members should contact their local DECUS office.



SD891205
 MR-4829-RA

Reader's Comments

We welcome your comments and suggestions regarding the *ALL-IN-1 Information Update*. They will help us in our continuing effort to improve the quality and usefulness of the Update. Please take a few minutes to complete this form and return it to us.

What do you think of the Update content?

- ☐ Too technical
- ☐ Not technical enough
- ☐ About right

How often do you refer to the Update?

- ☐ Occasionally
- ☐ Regularly
- ☐ Only to solve a problem

Is the Update of general use to you? (explain)

- ☐ Yes _____
- ☐ No _____

What other information would you like to see in the Update? _____

Please outline any errors you have found in the Update or any suggestions for improvement. _____

(Attach additional pages, if necessary)

Additional comments: _____

How many years experience do you have in the following?

- _____ ALL-IN-1
- _____ EDT and/or WPS editor
- _____ WPS-PLUS editor
- _____ Other office automation systems
- _____ Other word processing systems

I am/am not* willing to have you contact me by mail/telephone* regarding my comments and suggestions.

Name: _____

Title: _____

Company: _____

Street: _____

City: _____ State: _____ Zip Code: _____

Country: _____ Telephone: _____

*Delete as applicable.

Do Not Tear — Fold Here and Tape

digital™



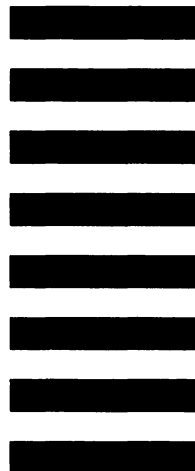
No Postage
Necessary
if Mailed in the
United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 33 MAYNARD, MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

Susan A. Thompson
Editor, ALL-IN-1 Information Update
Digital Equipment Corporation
Corporate User Publications (MR01-3/LP7)
P. O. Box 1001
Marlborough, MA 01752-9840



Do Not Tear — Fold Here and Tape